UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/628,959	07/28/2003	Eitan Hefetz	34874-020 UTIL	6174

64280 7590 08/10/2007
MINTZ, LEVIN, COHN, FERRIS, GLOVSKY & POPEO, P.C.
9255 TOWNE CENTER DRIVE
SUITE 600
SAN DIEGO, CA 92121

EXAMINER	
PATEL, MANGLESH M	
ART UNIT	**PAPER NUMBER**
2178	
MAIL DATE	**DELIVERY MODE**
08/10/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

[illegible]

Office Action Summary	Application No.	Applicant(s)	
	10/628,959	HEFETZ ET AL.	
	Examiner	Art Unit	
	Manglesh M. Patel	2178	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 14 May 2007.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-25 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-25 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This FINAL action is responsive to the amendment filed on 5/14/2007.

2. Claims 1-25 are pending. Claims 1, 6, 10, 14, 18 and 21 are the independent claims.

Withdrawn Rejections

3. The 35 U.S.C. 103(a) rejections of claims 1-24 with cited reference of Yu U.S. Pub 2004/0090458 have been withdrawn in light of the amendment.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Velonis (U.S. 6,772,408, filed Nov 22, 2000).

Regarding Independent claim 1, Velonis discloses a method comprising:

- Providing a design-time translator and a run-time translator that both correspond to a defined page element;
- During design-time for a page, invoking the design-time translator for a page template including the defined page element having one or more content components, said design-time invoking resulting in the defined page element in the page template being translated into a <u>design-time</u> representation of the one or more content components in the page <u>, the design-time representation being rendered in accordance with a layout of a container for the components</u>; and
- During run-time for the page, invoking the run-time translator for the page template, said run-time invoking resulting in the one or more content components being obtained and the defined page element in the page template being translated into a <u>run-time</u> presentation of

the obtained one or more content components <u>in accordance with the layout of the container</u>.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 2, which depends on claim 1, Velonis discloses wherein said invoking the design-time translator further results in presentation of a WYSIWYG layout editor using the <u>design-time</u> representation of the one or more content components in the page.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 3, which depends on claim 2, Velonis discloses wherein the said invoking the design-time translator further results in client-side scripting components being included in the design-time representation to form at least part of the WYSIWYG layout editor and enable adding a content component to a content container using a drag-and-drop action.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 4, which depends on claim 2, Velonis discloses wherein the page template comprises a portal page template, and the WYSIWYG layout editor comprises a WYSIWYG portal page layout editor.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 5, which depends on claim 4, Velonis discloses wherein the defined page element comprises a custom Java Server Page tag and the design-time translator and the run-time translator comprise Java Server Page tag handlers for the custom Java Server Page tag, and wherein the run-time translator obtains portal dynamic content according to the portal page template and the design-time translator does not.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Independent claim 6, Velonis discloses an article comprising a machine-readable medium storing instructions operable to cause one or more machines to perform operations comprising:

- During design-time of a portal page, translating a placeholder in a portal template into a design-time representation of a container designed to present portal dynamic content associated with the placeholder, and presenting a WYSIWYG portal layout editor using the design-time representation of the container designed to present the portal dynamic content , the run-time presentation being presented in accordance with the layout of the container;
- During run-time of a portal page, obtaining the portal dynamic content from a dynamic content source, and translating the placeholder in the portal template into a presentation of the container and the obtained portal dynamic content, the run-time presentation being presented in accordance with the layout of the container.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design.

Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 7, which depends on claim 6, Velonis discloses wherein translating the placeholder during design-time comprises adding code enabling editing of the portal page, the added code forming at least part of the WYSIWYG portal layout editor.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container

content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 8, which depends on claim 7, Velonis discloses wherein the added code comprises client-side scripting that enables addition of a content component to a content container in the portal page using a drag-and-drop action.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 9, which depends on claim 6, Velonis discloses wherein the placeholder comprises a custom Java Server Page tag, said translating the placeholder during design-time comprises invoking a design-time Java Server Page tag handler corresponding to the custom Java Server Page tag, and said translating the placeholder during run-time comprises invoking a run-time Java Server Page tag handler corresponding to the custom Java Server Page tag.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container

content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Independent claim 10, Velonis discloses a machine-implemented method comprising: selectively interpreting a portal page template based on a mode of operation, wherein the interpreting results in presentation of a design-time application operable to edit the portal page template if the mode of operation is design-time, and the interpreting results in presentation of a run-time application operable to interact with portal dynamic content if the mode of operation is run-time<u>, the portal page template including a container defining a layout of content, the content presented differently at design-time and run-time, and presentation of the content at design-time and run-time being in accordance with the layout.</u>

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are

integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 11, which depends on claim 10, the claim describes a method that contains the same limitations as claim 1 and is rejected under the same rationale.

Regarding Dependent claim 12, which depends on claim 11, Velonis discloses wherein said invoking the design-time translator further results in client-side scripting components being included in the representation to form at least part of the design-time application and enable adding a content component to a content container in the portal page template using a drag-and-drop action.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 13, which depends on claim 11, the claim describes a method that contains the same limitations as claim 5 and is rejected under the same rationale.

Regarding Independent claim 14, the claim describes an article that contains the same limitations as claim 10 and is rejected under the same rationale.

Regarding Dependent claim 15, which depends on claim 14, the claim describes an article that contains the same limitations as claim 1 and is rejected under the same rationale.

Regarding Dependent claim 16, which depends on claim 15, the claim describes an article that contains the same limitations as claim 12 and is rejected under the same rationale.

Regarding Dependent claim 17, which depends on claim 15, the claim describes an article that contains the same limitations as claim 5 and is rejected under the same rationale.

Regarding Independent claim 18, Velonis discloses a portal system comprising:

- A WYSIWYG portal layout editor that uses a selectively interpreted portal page template to reveal a WYSIWYG layout context for portal dynamic content without obtaining the portal dynamic content, the portal page template including a container defining a layout of content displayed differently at design-time and run-time in accordance with a first tag handler and a second tag handler;

- the first tag handler implementing a first custom action for a custom tag during portal design-time, wherein the WYSIWYG portal layout editor uses the first tag handler with the selectively interpreted portal page template to facilitate editing of the selectively interpreted portal page template , content of the first tag handler being presented in accordance with the layout;
- the second tag handler implementing a second custom action for the custom tag during portal run-time, wherein the portal system uses the second tag handler during portal run-time to obtain and reveal the portal dynamic content, the portal dynamic content of the second tag handler being presented in accordance with the layout.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container

content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 19, which depends on claim 18, Velonis discloses wherein the first tag handler interprets the portal page template by including client-side scripting that enables addition of a content component to a content container in the portal page template using a drag-and-drop action

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 20, which depends on claim 18, the claim describes a system that contains the same limitations as claim 5 and is rejected under the same rationale.

Regarding Independent claim 21, Velonis discloses a system comprising: means for building a portal layout template that governs generation of a portal presentation having dynamic run-time content, wherein the means for building includes means for revealing the portal presentation as governed by the layout template during design of the layout template, without running the dynamic run-time content <u>, the layout template including a container defining a layout of content, the content displayed differently at design-time and run-time, and presentation of the content at design-time and run-time in accordance with the layout.</u>

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 22, which depends on claim 21, Velonis discloses wherein the means for revealing the portal presentation includes means for facilitating client-side editing of the portal layout template.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are

integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 23, which depends on claim 1, Velonis discloses wherein the during design-time comprises a period during which editing for the page is supported and the during run-time comprises a period during which editing for the page is supported and the during run-time comprises a period during which editing of the page is not supported.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are

integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 24, which depends on claim 1, Velonis discloses wherein the design-time translator is part of a WYSIWYG layout editor, and the run-time translator is part of the run-time system that supports presenting the page without supporting editing of the page.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design. Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

Regarding Dependent claim 25, which depends on claim 1, Velonis discloses wherein the design-time translator is a WYSIWYG layout editor and changes to the layout of the container at design-time with the WYSIWYG editor are reflected in the layout of the design-time representation and the run-time presentation.

Velonis discloses the use of fidget's and non-fidget information that reside has server side components (see column 1, lines 15-40). He further describes that the content in a container or the fidget/non-fidget information generate at run-time and include Java Server pages (JSP) (see column 2, lines 18-67). Thus describing that at run-time the scripts defined in a JSP are executed to render the content components which include fidget/non-fidget information. The design-time translator and run-time translator is the parsing done by the browser that includes the defined page element as the JSP syntax at design-time and run-time, thus allowing the display of the content (see column 5, lines 5-30). Velonis fails to explicitly teach the rendering of content from a container at design-time. Velonis explains in column 4, lines 50-67 & column 5, lines 1-67 that fidgets are represented at design-time and run-time. At run-time it is represented by the fidget component itself, however he states in column 23, lines 10-32, that the JSP specification describes design tools that enable deployers to create and edit **JSP's visually, without having to worry about syntax** within a WYSIWYG editor. Further describing that the Fidgets are integrated with the JSP design tools to allow the rendering of fidgets. Thus Velonis **provides a reasonable suggestion** by showing that JSP are designed visually using a WYSIWYG editor commonly used during design-time. Thus JSP is typically executed during run-time to provide content components. Thus replacing placeholders in the script represented as page tags with content from the container. However Velonis shows how his fidgets are integrated into the visual design editing (WYSIWYG) component environment. The skilled artisan would also realize that Drag and drop action is a functionality of WYSIWYG editors, thereby allowing visual design.

Furthermore the content is presented differently at design-time and run-time because although the container content is rendered visually in editing or design phase not all content can be viewed such has run-time content which is generally not determinable until run-time. Therefore at the time of the invention it would have been obvious to one of ordinary skill in the art to allow visual design of JSP's that include rendering of content components in a WYSIWYG editor. The motivation for doing so would have been to save time by enabling visual design associated with JSP language for the novice programmer.

It is noted that any citation [[s]] to specific, pages, columns, lines, or figures in the prior art references and any interpretation of the references should not be considered to be limiting in any way. A reference is relevant for all it contains and may be relied upon for all that it would have reasonably suggested to one having ordinary skill in the art. [[See, MPEP 2123]]

Response to Arguments

6. Applicant's arguments filed 5/14/2007 have been fully considered but are moot in view of the new grounds of rejections.

Conclusion

References Cited

7. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Kitain et al (U.S. 7,246,134) discloses "System And Methods For Tag Library Generation"
- Foley et al. (U.S. 5,706,502) discloses "Internet-Enabled Portfolio Manager System And Method"
- Fukuda et al. (U.S. 7,246,041) discloses "Computer Evaluation Of Contents Of Interest"
- MyEclipse, MyEclipse Visual JSF Designer Quickstart, 2007, MyEclipse, pgs 1-25
- Javabeat, Java Server Faces (JSF), 2007, Javabeat, pgs 1-46

8. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Manglesh M. Patel whose telephone number is (571) 272-5937. The examiner can normally be reached on M, W 6 am-3 pm T, TH 6 am-2pm, Fr 9am-6pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Stephen S. Hong can be reached on (571) 272-4124. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Manglesh M. Patel
Patent Examiner
August 4, 2007

[illegible]

[illegible]
CESAR PAULA
PRIMARY EXAMINER